forums.microsoft.com/MSDN/ShowPost.aspx?PostID=1218417&SiteID=1 - 51k -
Cached - Similar pages

Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**

---

query extensions to exclude Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

---

Google Home - Advertising Programs - Business Solutions - About Google

©2007 Google

Facilitating complex Web queries through visual user interfaces and query relaxation

Wen-Syan Li and Junho Shim

C&C Research Laboratories, NEC USA, Inc.,
110 Rio Robles, M/S SJ100, San Jose, CA 95134, U.S.A.
wen@ccrl.sj.nec.com and jshim@ccrl.sj.nec.com

Abstract

The World Wide Web can be viewed as a collection of multimedia documents in the form of HTML pages connected through hyperlinks. We have designed and implemented a Web query system, WebDB, to support more comprehensive database-like query functionalities. WebDB supports queries on not only document level information (e.g. title, URL, keywords) but also intra-document structures (e.g. tables, forms, and images) and inter-document linkage information (e.g. URLs and anchors) To provide higher usability for a system with such functionalities, we have designed a novel visual user interface, WebIFQ (Web In-Frame-Query), to assist users in specifying queries and visualizing query criteria including document metadata, structures, and linkage information. WebIFQ automatically generates corresponding query statements for WebDB. As a result, users are not required to be aware of underlying complex schema design and language syntax. WebDB supports automated query relaxation to include additional terms related by semantic or co-occurrence relationship. Alternatively, WebIFQ can facilitate users to reformulate queries perpetually in an interactive mode.

Keywords:

Search and indexing techniques; Information retrieval and modeling; Human–computer interaction; User interface

1. Introduction

The World Wide Web can be viewed as a collection of multimedia documents (pages) connected through hyperlinks. We categorize information available on the Web as follows: (1) Document information, such as type, size, last modified date, URL, page title, and keywords; (2) inter-documentation information including links from/to/within a page and anchor labels. Links within a page are through so-called *labels*; and (3) intra-document information, such as forms, images, tables, and links.

With all these three types of information, more complex queries can be supported. A more comprehensive query, such as "retrieve all pages, modified after 1997, which are linked from www.nba.com with depth of 10, sort the results by their URLs, and remove duplicate pages", can be supported. This query can be used as a *spider* to collect documents from www.nba.com and to organize the results. The query "retrieve all pages which have links to www.nba.com, group them by country of URL locations, and display the numbers of pages for each country" can be viewed as using a query to conduct a market survey for geographic locations of NBA fans.

We have developed a Web query system, WebDB, to support advanced Web search functionalities. WebDB extracts the Web structure and HTML document internal structure to allow search on Web document structures, such as forms and tables, as well as inter-document linkage information, such as

links and anchors. WebDB also supports multimedia search capabilities through a multimedia database system, SEMCOG [9]. In other words, WebDB views the Web as a huge hypermedia database and provides full-fledged database-like query functionality.

In addition, WebDB provides high usability through strong emphasis on computer human interaction aspect. WebDB features a visual query interface and a query generator, WebIFQ (Web In-Frame-Query), to assist users in formulating complex Web queries. WebIFQ visualizes query criteria as query specification processes. Users have a clear overview of query criteria, including linkage and intra-document structures. WebDB supports various automated query relaxation schemes. Alternatively, users can interact WebIFQ for query refinement, relaxation, and reformulation.

We illustrate these features in Fig. 1. Here, a user wants to retrieve all Web pages containing both an HTML form and the keyword "multimedia" (or other terms related by semantic similarity or co-occurrence relationship) which have links to the NEC Web sites in *www.ccrl.neclab.com* within link depth of 3. The URLs of these NEC pages which are linked by these outside pages are to be projected.

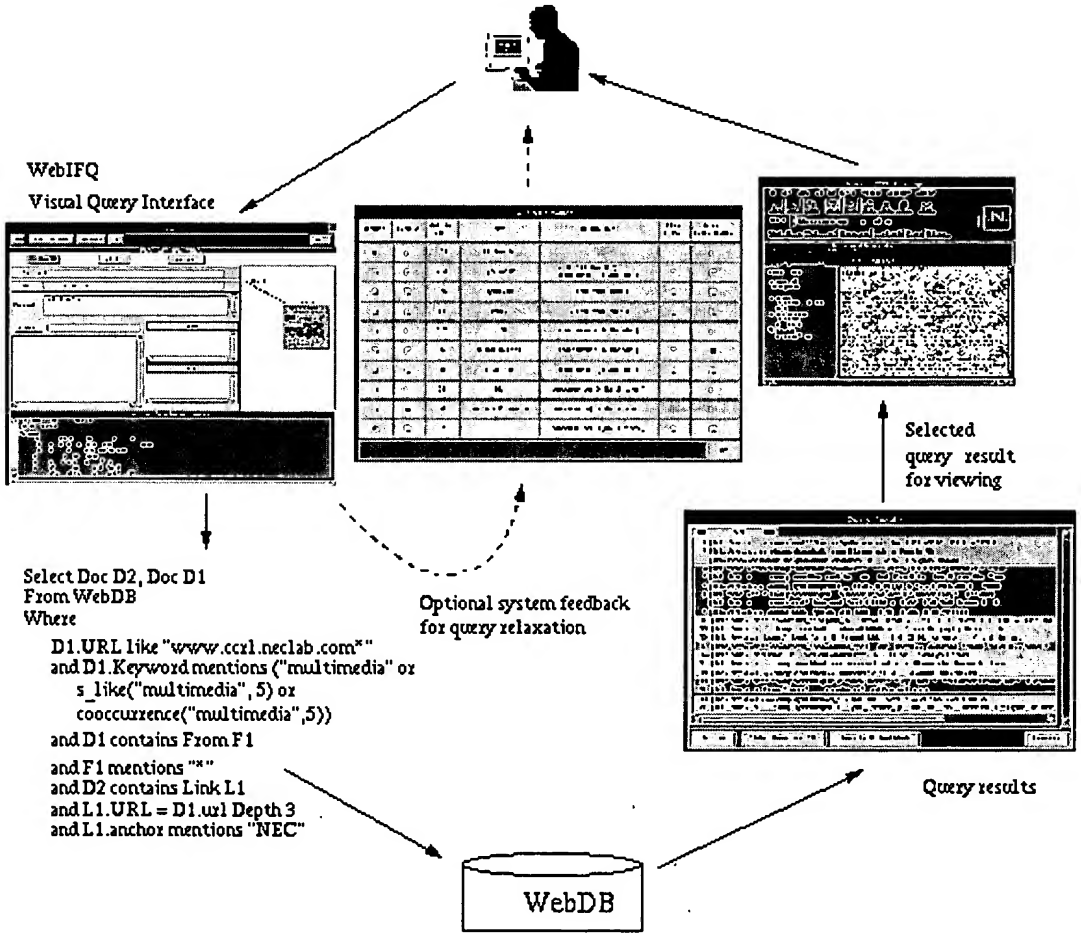


Fig. 1. Querying Web documents in WebDB.

Figure 1 shows that users use a visual query interface, WebIFQ, to specify queries, rather than using the complex query language directly. The data modeling in WebDB is based on the object-relational concept and the above query can be specified using WQL (Web Query Language), based on SQL3, as show in Fig. 1. Note that the projected string "⟶" is for the purpose of output presentation and

mentions is a string matching function for a set of strings, such as a keyword list.

Keywords are one of the most important and frequently used query criteria. WebDB supports two types of automated query relaxation through: `s_like` function for semantically related terms and `cooccurrence` function for terms related by co-occurrence relationship. WebDB also allows users to relax, reformulate, or refine queries through interactions with WebIFQ as shown in the centre of Fig. 1. Users can include or exclude particular terms for search or request additional related terms for these terms perpetually. The corresponding query statements are automatically generated by WebIFQ. The query statement in WQL is then processed by the WebDB query processor. The result of the above query may be as follows:

http://www.ece.nwu.edu/~ shimjh ⟶ http://www.ccrl.neclab.com/Anecdote
http://www.ece.nwu.edu/~ shimjh ⟶ http://www.ccrl.neclab.com/nec_sj/
http://www.ece.nwu.edu/~ acura ⟶ http://www.ccrl.neclab.com/forum97/

The result is presented to the user through a browser, such as Netscape Navigator. The user can click on any of the presented URLs to browse a particular page or can save these URLs as bookmarks for later use. WebDB also supports *slide-show* functionality, i.e. automated display of all pages or selected pages (e.g. first 10 pages).

The rest of this paper is organized as follows: We first review related work. In Section 3, we present an overview of the Web modeling schemes and query language design in WebDB. In Section 4, we present the design and operations of WebIFQ using some example queries. In Section 5, we present the system architecture of WebDB and indexing schemes to support `s_like` and `cooccurrence` functions. We give our conclusions in Section 6.

2. Related work

Most information retrieval engines for the Web provide search capabilities only by keyword or phrase and criteria combinations using Boolean expressions without considering the Web structure and multimedia components. Examples of these systems include Altavista [1], InfoSeek [2], Yahoo [3], and Excite [4]. Altavista is distinct as it includes a query refinement interface called *Live Topic*. WebDB supports query refinement as well as query relaxation and query reformulation.

WebSQL [5] is a project at University of Toronto to develop a Web query facilitation language. It views the Web as a table of documents, in which URL, Title, Type, Last Modified Date are treated as columns. WebSQL extends standard SQL by adding information related to Web documents, such as URL and Title, as column names for queries. Some user-defined functions, such as "mentions", are supported for more fuzzy textual string matching. The query interface provided for WebSQL is form-based, as opposed to the visual query interface and query generator provided by WebDB.

WebLog [6], developed at Concordia University, is a declarative language for Web queries based on SchemaLog. It is intended to be a more complete language to support both query and result rendering formatting. No implementation of WebLog has been reported. TSIMMIS [10] is a project at Stanford University to support query heterogeneous information resources. TSIMMIS is similar to WebLog, but it implements many pre-defined queries for information retrieval so that users need not pose complex queries directly. But, this restricts searches using limited pre-defined queries.

W3QS (WWW Query System) [7] at Technion (Israel Institute of Technology) is a project to develop a high level SQL-like Web query language, W3QL, which views the Web as an ultra large database.

W3QL addresses both structure and content. W3QS allows users to specify file types and file names using Perl regular expressions for search. W3QS supports queries on the Web structure by specifying a starting page, a search domain, and the depth of links. In comparison, WebDB also allows users to specify queries with arbitrary Web structures; it is not limited to one link-in or one link-out. Moreover, WebDB features a more user-friendly query interface and supports query relaxation.

HyperFile [8] is a data and query model for hypertext documents. It introduces sophisticated modeling scheme and focuses query processing technique. Compared with HyperFile, WebDB is a query system for hypermedia documents on the Web. Additionally, WebDB supports additional functionalities, such as a visual query interface and query relaxation, to provide higher usability.

3. Web modeling

We view and model Web as a labeled directed graph $G_{web} = (V_{web}, E_{web})$, where the vertices ($V$) denote the pages and the edges (E) denotes the hyperlinks between these pages. The vertices are labeled by the URLs of the pages and other document level information, including title, URL, content length, data types, last modified date, and keywords. We further model each vertex, $n_i \in V_{web}$, as a compound object which consists of text, images, tables, and forms. The edges are links from source pages to destination pages and are labeled by the *descriptive text*: anchors.

To model the Web, we take the approach of object-relational modeling. The intra-document structures are modeled using the object-oriented model while the query language is based on SQL3 (an extension of a relational query language SQL). The Web modeling in WebDB is illustrated in Fig. 2 and is as follows:

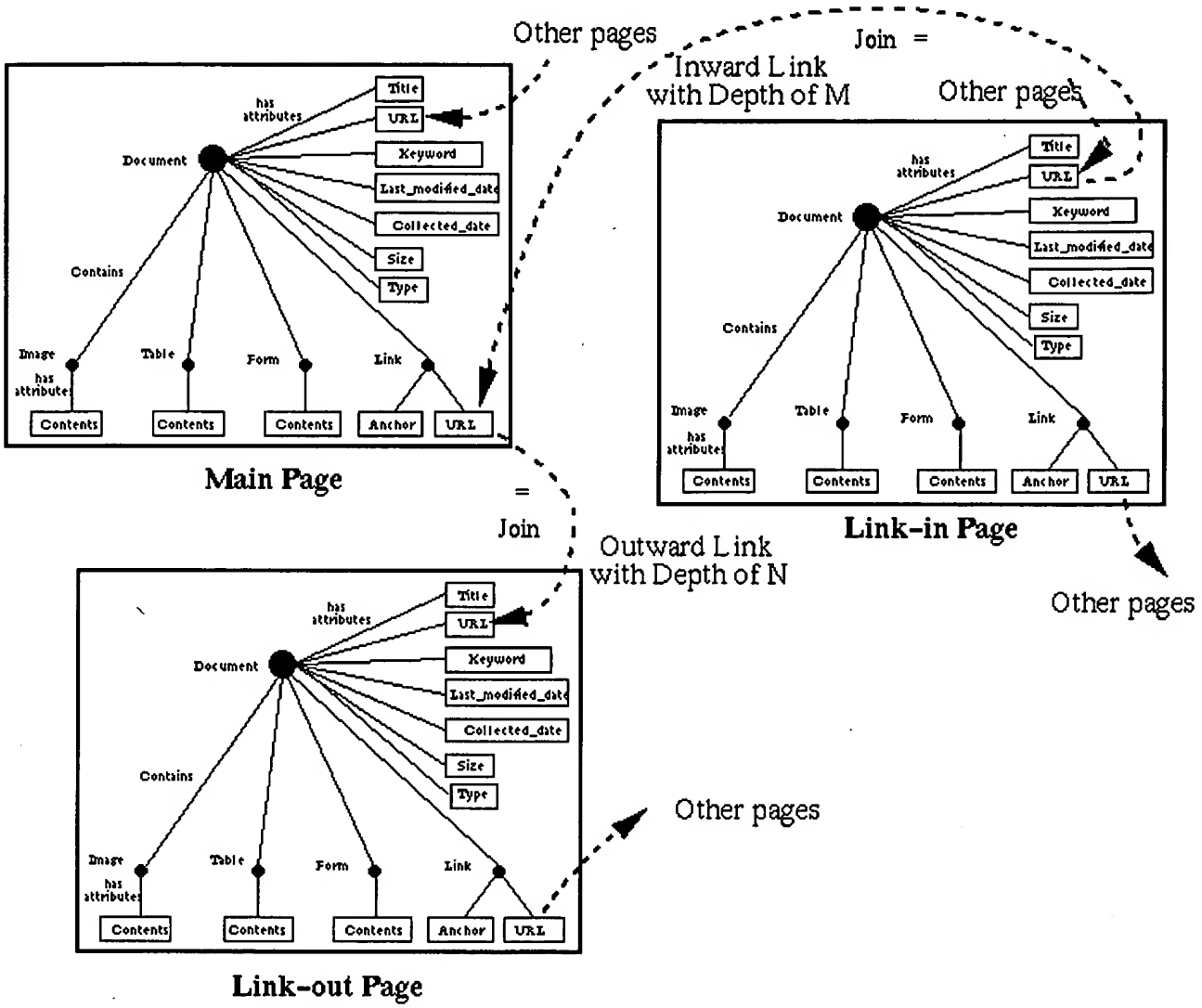


Fig. 2. Web modeling in WebDB.

- A Web document, *Doc*, is modeled as a compound object with a hierarchical structure. Document level information, such as title, URL, content length, file types, last modified date, collected date, and keywords, are the attributes of *Doc* object.

- Intra-document structures are modeled as sub-objects of *Doc*, including *Form*, *Image*, *Table*, and *Link*. The relationship between *Doc* and the sub-objects is *contains*. Sub-objects also have their own attributes. The attribute for *Image* is image metadata while the attribute for *Form* and *Table* are contents contained in forms and tables. Formally we represent their attributes as *Image.Metadata*, *Form.Contents*, and *Table.Contents*.

- Inter-document information is represented by *Link*, which is a sub-object of *Doc*. *Link* has two attributes: *URL* for the destination URL and *Anchor*. The inter-document link is modeled *implicitly* through `join` operations on *Doc1.Link.URL* and *Doc2.URL*, where *depth* is a parameter for `join` operations, defining the number of `join` operations to be performed recursively. The intra-document link (i.e. label) is modeled through *Join* operations on $Doc_i.Link.URL$ and $Doc_i.URL$.

By viewing objects as entities and links as relations, we map the modeling representation in Fig. 2 to the Entity-Relational (ER) model to design the query language. Since we model Web documents as compound objects with structures, our query language is based on SQL3, an extension of the traditional SQL. In the next section, we show how to match Web queries to WebIFQ specifications. By viewing objects as entities and links as relations, we map the modeling representation in Fig. 2 to the Entity-Relational (ER) model for the WQL language design. Since we model Web documents as compound objects with structures, we extend the traditional SQL with the following functionalities:

- **Traversal of the intra-document structure:** The intra-document structure traversal is by way of the predicate *contains*.

- **Traversal of the Web (inter-document links):** The Web structure, on the other hand, is modeled through `join` operations on documents' hyperlinks. Traversal of the Web from page *Doc_x* to page *Doc_y* through a link with depth of 2 is by way of the following `join` operations:

  Doc_x.Link.URL = Doc_y.URL
  or
  (*Doc_x.Link.URL = Doc_z.URL* and *Doc_z.Link.URL = Doc_y.URL*)

- **Similarity-based image matching:** WebDB provides an *i_like* (image like) predicate to perform image matching. The image matching functionality is carried out by an image database SEMCOG and users can specify image related queries using the IFQ visual query interface. Detail information of SEMCOG and IFQ is available in [9].

4. WebIFQ query interface

4.1. Query specification

WebDB features a visual query interface, WebIFQ, to assist users in specifying queries. There are two windows in WebIFQ: `Search Specification Window` and `WQL Window`. As the name of Web In-Frame-Query implies, users pose queries in a frame, `Search Specification Window`, in a drag-and-drop fashion. The corresponding query statements are automatically generated by the system in `WQL Window`. As a result, users are not required to be aware of complex underlying schema design and language syntax.

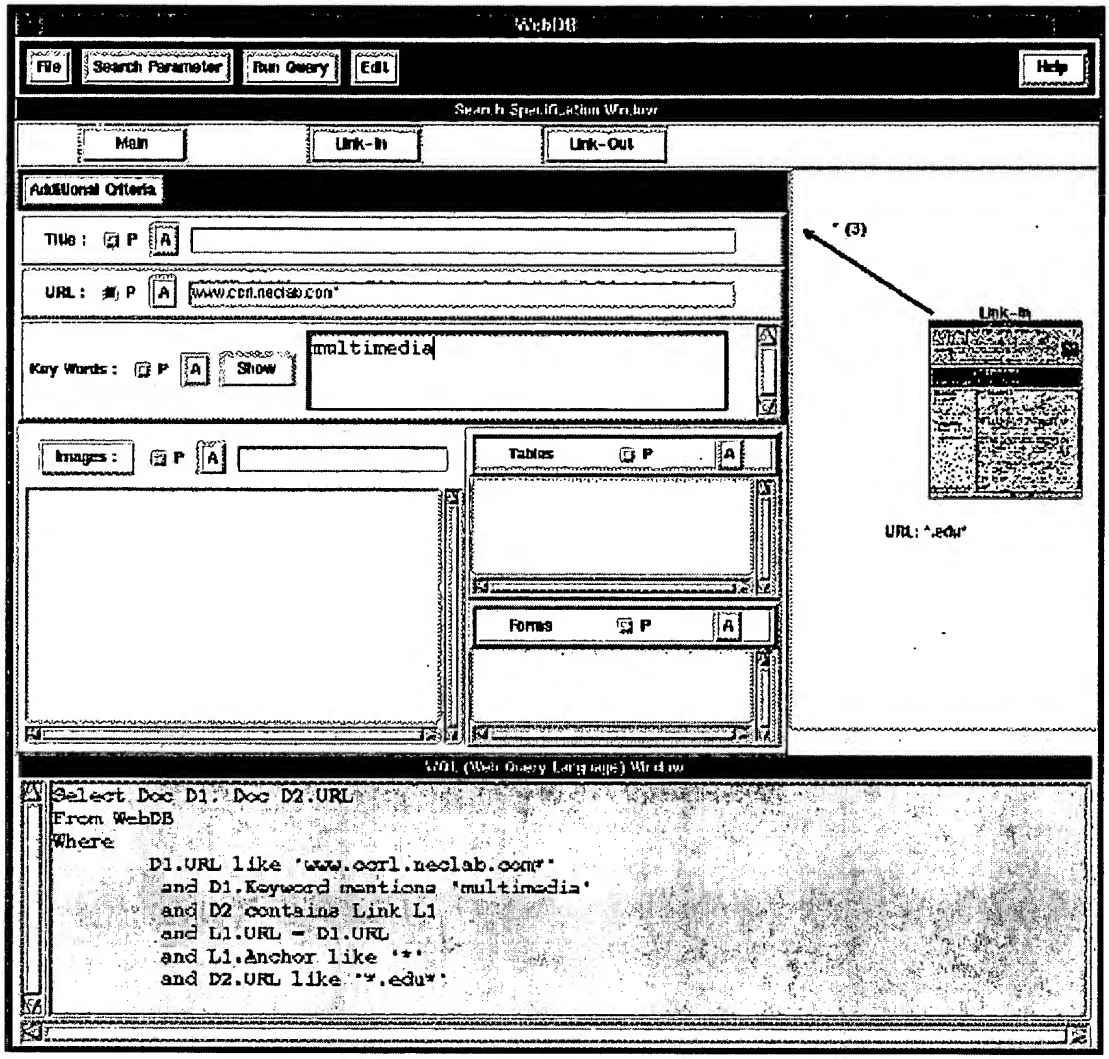


Fig. 3. Query specification using WebIFQ (main window view).

There are three types of windows, namely, main, link-in, and link-out windows. WebIFQ allows users to switch between these windows to specify query criteria associated with each window by clicking the `Main`, `Link-in`, and `Link-out` buttons at the top of `Search Specification Window`. When users specify query criteria in one window, the system shrinks other windows but display their summarized query criteria.

Figure 3 shows the query specifications from the main window view while the link-in window is shrunk: the user specifies the search criteria for `URL`, `Keywords`, and `Form`. After the user clicks the `Link-in` button, `Search Specification Window` switches from the main window view (Fig. 3) to the link-in window view, in which the main window is shrunk while the link-in window is in the normal size.

To specify the criteria associated with linkage, users click on the link between the main window and link-in or link-out windows. A window will pop up to allow the users to specify the anchor and depth conditions. WebIFQ visualizes the linkage relationship between the main window and the link-out window as well as the anchor and depth conditions.

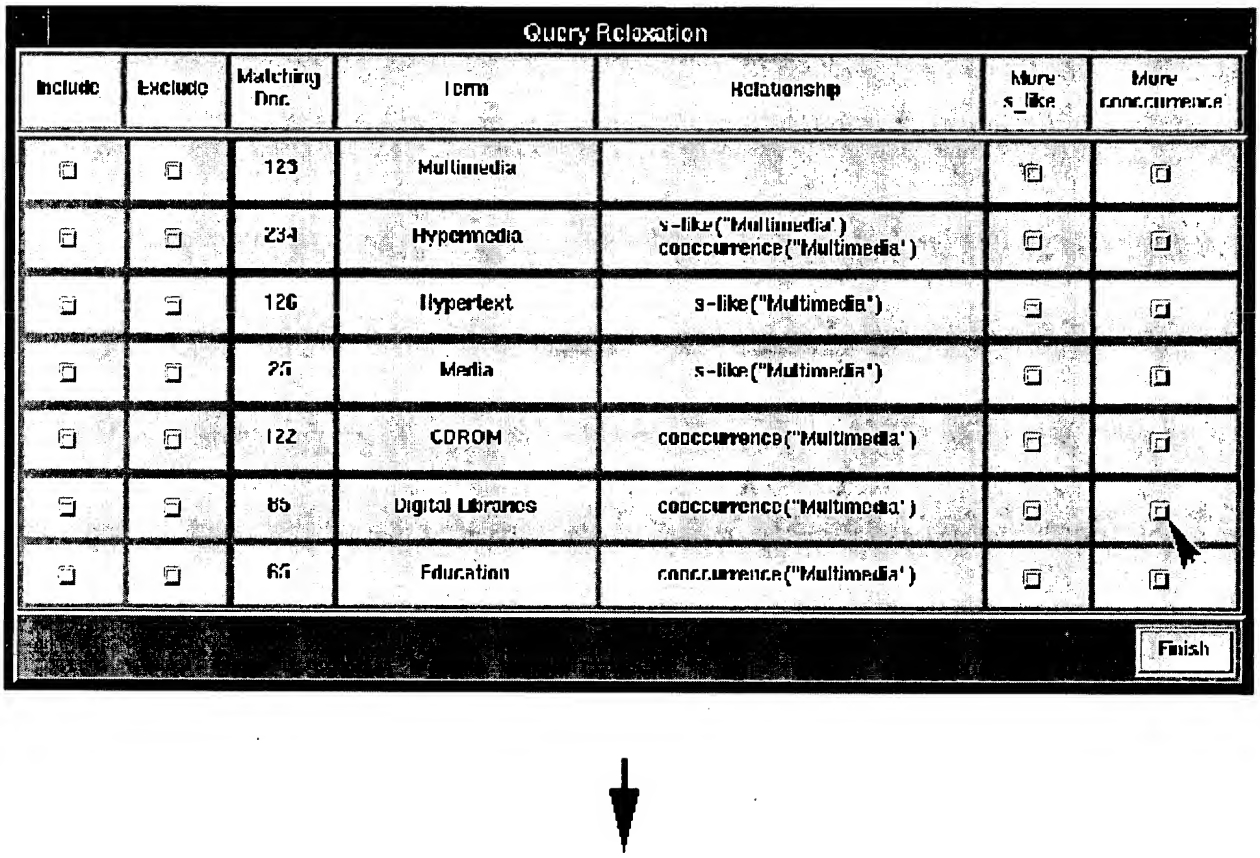
4.2. Query relaxation

Keywords is one of most important and most frequently used query criteria. WebDB supports query

relaxation by including additional terms related by semantic similarity or co-occurrence relationship. The details of indexing schemes of these two functions are given in Section 5. For the keyword criteria:

Multimedia or s_like("multimedia",3) or cooccurrence("multimedia",4)

the system relaxes the query criteria by automatically extending "multimedia" with other related keywords for query processing: three related terms by semantic similarity and four additional terms by cooccurrence relationship. Alternatively, WebDB also allows users to relax, reformulate, or refine queries through interactions with WebIFQ. User can click on the Show button, next to the keyword field, to see the alternative terms. In this example, the user clicks on the Show button, a window shown at the top of Fig. 3 then pops up to allow users to display terms related by `s_like("multimedia",3)` and `cooccurrence("multimedia",4)`. Users can further relax these terms.



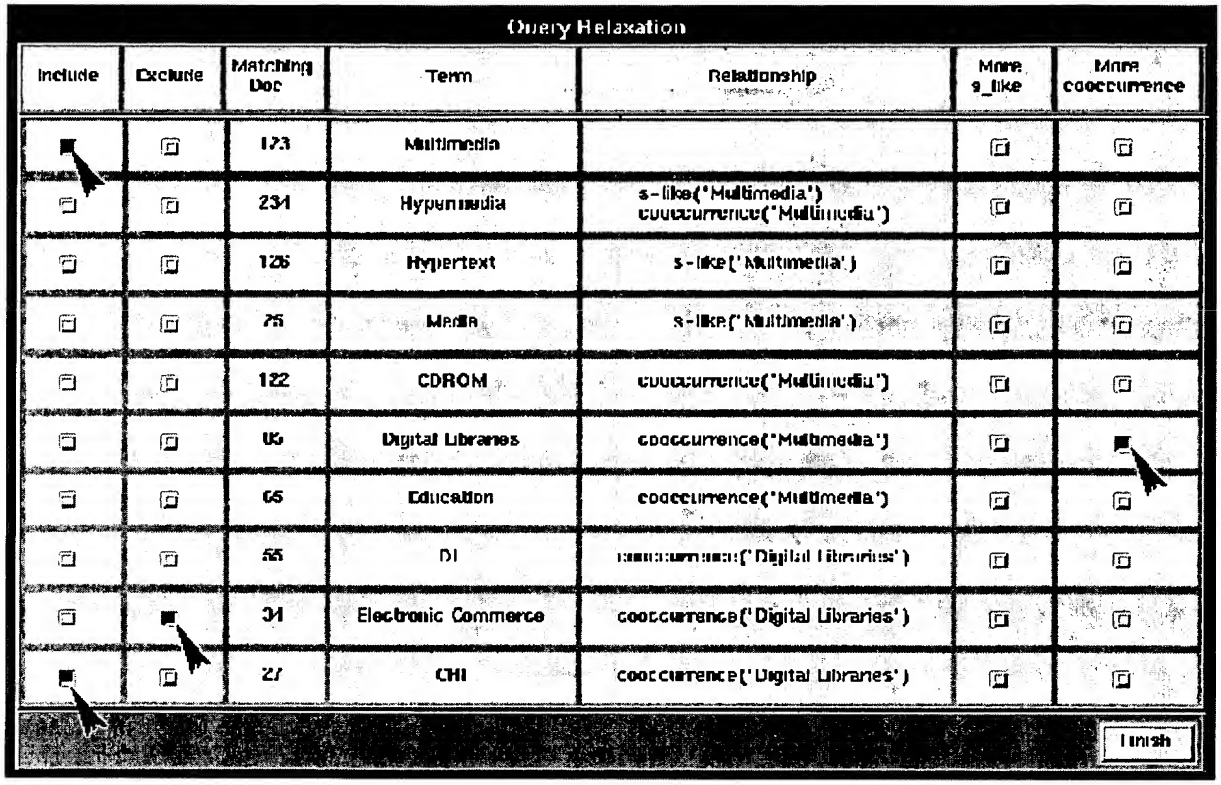


Fig. 4. Perpetual query relaxation interface window.

In this example, the user selects "multimedia" and requests the system to provide additional terms related to "Digital Libraries" by co-occurrence relationship (indicated with an arrow). Currently, the system is set to provide 3 additional terms each time. As a result, "DL", "Electronic Commerce", and "CHI" are presented in the bottom of Fig. 4. The user then includes "CHI" and excludes "Electronic Commerce" for search. Note that, the system also shows users the number of documents which contain a particular term. After the query relaxation and reformulation, the new keyword query criteria is as follows:

Multimedia or CHI and not("Electronic Commerce")

For the interactive mode of query reformation, there are two types of implementation we are considering for different network capacity. In a network environment with a high bandwidth, the interaction between users and the system is conducted in real time. In a network environment with a low bandwidth, the system sends a set of terms in advance. In this query example, the system may send all possible terms and selectivity for up to four levels of query relaxation interaction at once in advance: 1,520 terms: $1 + 3 + 4 + (3 \times 6^3) + (4 \times 6^3)$ (i.e, 3 s_like terms and 4 cooccurrence terms for 3 additional levels) and their selectivity. The cost of sending 1,520 terms is not expensive. This scheme can reduce future communication setup time for further interaction between clients and WebDB.

5. Design and implementation

The system architecture of WebDB consists of four major components: *Document Parser* and *Document Indexer* are involved in the indexing step and *Query Pre-processor* and *Query Processor* are involved in the query processing step. The indexing process is followed by the query processing step. We next show the functionalities of each component and focus on term extraction and indexing schemes for query

relaxation functionalities described in Fig. 4.

5.1. Document parser

To perform the document gathering task, we have explored Harvest [11], Web search engines (e.g. AltaVista), and so-called spiders to gather Web pages. Currently we utilize Harvest to perform document gathering for specific domains (e.g. www.nba.com). We also use Harvest's parser to extract document level metadata, including URL, keyword, title, last modified date, type, document type, and size. To parse intra- and inter-document information, we have implemented a parser using `Perl` to extract this information.

Parsing has a great deal of impact on the quality of metadata extracted and query results consequently. Harvest extracts keywords based on whether or not a word is highlighted by special typeface tags, such as boldface, italic, or underlined. To improve the quality of parsing, *Document Parser* performs additional *stemming* to remove words in the forms of verb and adverb by consulting *Terminology Dictionary* (currently WordNet [12] is used). WordNet is a Lexical Database for English. In WordNet, English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept.

Document Parser also transforms words in the plural form to their singular form. Additional improvements can be made by extracting "terms" rather than "keywords". For example, Michael Jordan in an HTML document is identified by Harvest/Essence extraction system as two keywords, michael and jordan. For Taxi driver, the keywords are identified as "taxi" and "driver". These two extraction results are not proper since "Jordan" may be matched with the country "Jordan" and "driver" may be matched with a golf "driver".

We are implementing and testing a new parser which further explores sentence structures and examines word forms. The following rules are being added to the parsing procedure:

- A sequence of capitalized words are grouped as one single proper name. For example, Michael Jordan will be treated as one keyword, rather than two.
- Words in the form of noun or adjective before a noun should be grouped together as one keyword. For example, "taxi driver", "fast car", "golf shop" will be parsed as three terms, rather than six.

By applying these rules and consulting Wordnet, the parser can extract three terms, "Michael Jordan", "fast car", and "golf shop", from a highlighted sentence "Michael Jordan drives a fast car to a golf shop".

5.2. Document indexer

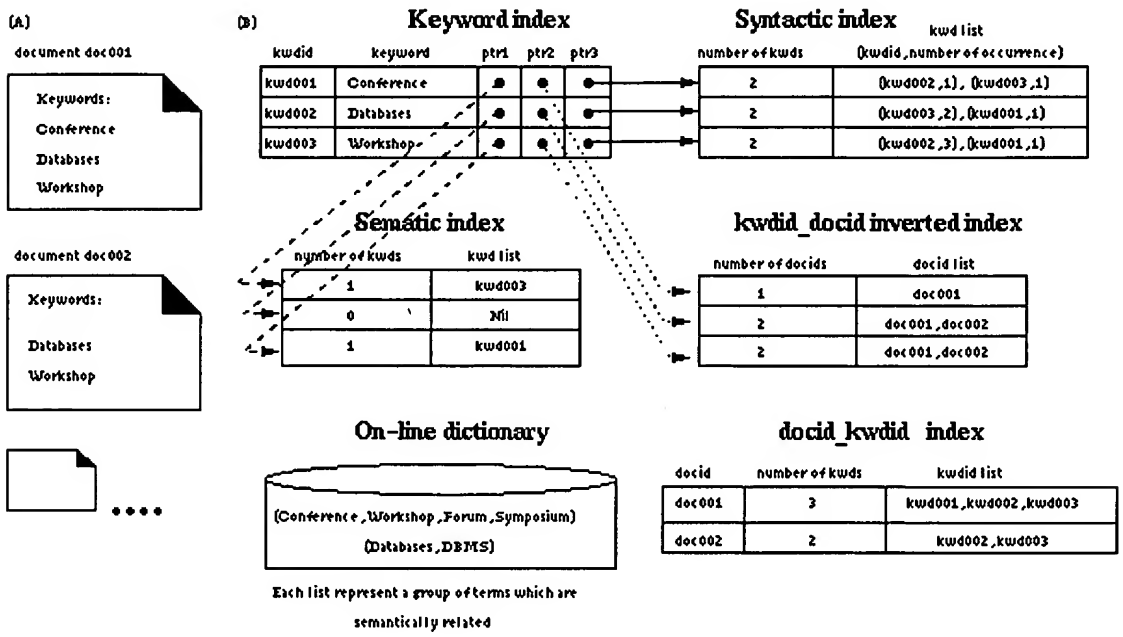


Fig. 5. Indexing scheme in WebDB.

Document Indexer is responsible for the following tasks:

- Building *Semantically Relevant Term Index* for terms which are semantically relevant: This index is used by the function *s_like* to relax keyword-based search criteria. For example, a query with the keyword *car* or terms semantically related by *s_like* may retrieve documents with *car*, *truck*, *bus*, or *sedan*. *Semantically Relevant Term Index* is built through consulting an on-line terminology dictionary; currently Wordnet is used. These semantically relevant terms are also used for system feedback to show users alternative terms for query reformulation. The index structures are shown in Fig. 5.

  In Fig. 5, we use two documents as examples to illustrate the indexing schemes in WebDB. Each document and keyword have their own identifiers. Two documents are identified as *doc001* and *doc002* and the keywords "conference", "databases", and "workshop" are assigned as *kwd001*, *kwd002*, and *kwd003* respectively. Based on `docid_kwdid index`, we can find keyword IDs for a given document. Based on `Keyword index`, we can find keywords for a given keyword ID. An inverted index for `docid_kwdid index`, `kwdid_docid inverted index`, is constructed to find all document IDs which contain a given keyword. Note that the values of the attribute `number_of_docids` are the document selectivity for keyword searches. These values are used for query optimization purposes and for showing users as system feedback as shown in Fig. 4.

  One keyword may be associated with other keywords by semantics or cooccurrence relationships. In this example, the keyword "conference" is associated with "workshop", "forum", and "symposium" by semantics. Since only the term "workshop" exists in other document (i.e. doc002), an entry `(1, kwd003)` is inserted into `Semantic index` for space saving purposes. However, if a user poses a query using the keyword "forum" or "symposium" which does not exist in the collection of WebDB, the system can associate documents with the keyword "conference" or "workshop" by consulting the on-line dictionary during query processing time.

- Building *Cooccurring Term Index*: Not all words used in documents can be found in a dictionary. Many of these words are so-called proper names. *Semantically Relevant Term Index* is used to

index semantically relevant terms, while *Cooccurring Term Index* is used to index "syntactically" relevant terms, based their frequency of appearance in the same document together.

This index is used by the function *cooccurrence* in relaxing search based on keywords. The *cooccurrence* function allows users to search documents containing a keyword or other keywords often appearing with this keyword on documents. For example, a relaxed query to find documents with the keyword "*Michael Jordan*" using the the function *cooccurrence* may retrieve documents with "*Michael Jordan*", "*NBA*", "*Chicago Bulls*", "*Scotty Pippen*", etc.

The cooccurrence frequency for two terms can be viewed as the selectivity for a query with these two terms. The second entry of `Syntactic index` indicates `kwd002` and `kwd003` co-occur in the same document in two occasions and `kwd002` and `kwd001` co-occur in the same document in one occasion. `Syntactic index` is used for query optimization as well as for system feedback to show users relevant terms for query reformulation as illustrated in Fig. 4.

- Constructing *Textual Metadata Database* for tables, forms, keywords, titles, links, anchors, and other document level metadata and *Image Metadata Database* for images. The textual information described in Section 3 is stored in *Textual Metadata Database*, while other non-textual information (i.e. image metadata) is stored in *Image Metadata Database*.

6. Conclusion

WebDB is an advanced Web query system based on object-relational concepts. It provides a query language based on SQL3 for access to document structures, Web linkage, and multimedia data in a uniform manner. We have demonstrated many useful applications of this system. To provide higher usability for a system with such functionalities, we have designed a visual user interface, WebIFQ (Web In-Frame-Query), to facilitate complex Web queries.

The contributions of this work include the follows:

- A Web query system with many advanced functionalities and high usability through strong emphasis on computer human interactions.
- WebIFQ assists users in specifying queries and visualizing query criteria including document metadata, structures, and linkage information.
- WebIFQ is a query generator for complex Web queries so that users are not required to be aware of underlying complex schema design and language syntax.
- WebIFQ supports automated query relaxation by both semantic similarity and cooccurrence relationships.
- WebIFQ can interact with users for query relaxation. In the interactive mode, users can see the relationships between terms, document selectivity for each term, and why they are related to the initial query criteria. Users can request additional terms and control how these additional terms are related (by semantic similarity or cooccurrence relationships).

References

1 AltaVista Technology, Inc. of California, information available at http://www.altavista.com/

2 Infoseek Corporation, information available at http://www.infoseek.com/

3 Yahoo Communications Corporation, information available at http://www.yahoo.com/

4 Excite Inc., information available at http://www.excite.com/

5 A. Mendelzon, G. Mihaila, and T. Milo, Querying the World Wide Web, *Journal on Digital Libraries*, 1(1): 54–67, 1997.

6 L.V.S. Lakshmanan, F. Sadri and I.N. Subramanian, A declarative language for querying and restructuring the World Wide Web, in: *Proceedings of 1996 IEEE RIDE-NDS*, New Orleans, USA, February 1996.

7 D. Konopnicki and O. Shmueli, W3QS: a query system for the World Wide Web, in: *Proc. of the 21th International Conference on Very Large Data Bases*, VLDB, 1995.

8 C. Clifton, H. Garcia-Molina, and D. Bloom, Hyperfile: a data and query model for documents, *VLDB Journal*, 4(1), March 1995.

9 W.-S. Li and K. Candan. SEMCOG: a hybrid object-based image database system and its modeling, language, and query processing, in: *Proc. of the 14th International Conference on Data Engineering*, February 1998 (to appear).

10 J. Hammer, H. Garcia-Molina, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom, Information translation, mediation, and mosaic-based browsing in the tsimmis system, in: *Proceedings of the 1995 ACM SIGMOD Conference*, San Jose, California, May 23–25, 1995.

11 C.M. Bowman, P.B. Danzig, D.R. Hardy, U. Manber, and M.F. Schwartz, The Harvest information discovery and access system, in: *Proc. of the 2nd International World Wide Web Conference*, October 27–29, 1995, pp. 763–771.

12 G.A. Miller, WordNet: a lexical database for English, *Communications of the ACM*, pp. 39–41, November 1995.

Vitae

Wen-Syan Li is currently a research staff member at Multimedia Software Department of Computers and Communications (C&C) Research Laboratories, NEC USA Inc. He received his Ph.D. in Computer Science from Northwestern University in December 1995. He also holds an MBA degree. His main research interests include multimedia/hypermedia/document databases, heterogeneous databases, object-relational database systems, and user interfaces.

Junho Shim received his M.S. degree in Computer Science from Seoul National University in Korea. He is currently a Ph.D. student at Northwestern University. His major interests include database system, multimedia database, client-server architecture, and WWW. The work described here was performed when the author visited C&C Research Laboratories, NEC USA Inc.

Sign in

Google

Web Images Video News Maps more »

query extensions to exclude terms in query Search Advanced Search Preferences

Web Results **1 - 10** of about **801,000** for **query extensions to exclude terms in query**. (**0.20** seconds)

Google SOAP Search API
Exclude Query Term, bass -music, You can **exclude** a word from your search by ... there can be no space between "filetype:" and the specified **extension**. ...
code.google.com/apis/soapsearch/reference.html - 100k - Cached - Similar pages

Searching Data - Beagle
... so you can search for properties by phrase, using wildcards, **exclude terms**, or provide optional **terms**. For example, the following **query** will return all ...
beagle-project.org/Searching_Data - 18k - Feb 28, 2007 - Cached - Similar pages

Google **Query** Syntax Reference
Exclude Query Term, bass -music, Sometimes what you're searching for has more ... Note: Multiple **extensions** can be included in a filtered search by adding ...
www.geocities.com/ian_springer/google_syntax.html - 21k - Cached - Similar pages

websphere application server information center
Comparison of EJB 2.0 specification and WebSphere **query** language ... Developing servlets with WebSphere Application Server **extensions** ...
publib.boulder.ibm.com/infocenter/wasinfo/index.jsp - Similar pages

BMW : gsok
Exclude Query Term, bass -music, You can **exclude** a word from your search by ... to include only documents with the **extension** specified immediately after. ...
www.beemernet.com/mx/gsok/list/ - 17k - Cached - Similar pages

SPARQL **Query** Results XML Format
Internet Media Type, File **Extension** and Macintosh File Type ... Changed html style of references to **terms** defined in SPARQL **Query** such as **Query** Variable, ...
www.w3.org/TR/2005/WD-rdf-sparql-XMLres-20050801/ - 37k - Cached - Similar pages

URIQA: The Nokia URI **Query** Agent Model
URIQA proposes an **extension** to web architecture used to indicate to a web server that it should resolve the specified URI in **terms** of knowledge about the ...
sw.nokia.com/uriqa/URIQA.html - 20k - Cached - Similar pages

GSAQuery.GSAQueryTerm
This class is required because the GSA **query term** supports a "special **term**" syntax. eg. when ... **exclude** documents with specified file **extensions** ...
gsa-japi.sourceforge.net/apidocs/net/sf/gsaapi/GSA**Query**.GSA**Query**Term.html - 28k - Cached - Similar pages

Facilitating Complex Web Queries through Visual User Interfaces ...
Users can include or **exclude** particular **terms** for search or request additional related **terms** for these **terms** perpetually. The corresponding **query** statements ...
www7.scu.edu.au/1936/com1936.htm - 35k - Cached - Similar pages

Microsoft Index Server Guide: **Query** Language
The **query** esc* matches the **terms** "ESC," "escape," and so on. ... it can be used only to **exclude** pages that match a previous content restriction. ...
www.state.wv.us/qrylang.htm - 43k - Cached - Similar pages

Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**

---

query extensions to exclude terms ir Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

---

Google Home - Advertising Programs - Business Solutions - About Google

©2007 Google

Sign in

Google

Web Images Video News Maps more »

query extensions to exclude Search Advanced Search Preferences

Web Results **1 - 10** of about **837,000** for **query extensions to exclude**. (**0.19** seconds)

websphere application server information center
Comparison of EJB 2.0 specification and WebSphere **query** language ... **Exclude** list assembly settings · **Extension** MBean collection ...
publib.boulder.ibm.com/infocenter/wasinfo/index.jsp - Similar pages

sub-**query** trouble is making me so mad! - Dreamweaver MX and ...
... UltraDev and Fireworks MX developers! Here you can find tutorials, **extensions**, faqs en much more! ... The current **query** to include or **exclude** is: ...
www.dmxzone.com/forum/go?29969 - 21k - Cached - Similar pages

SourceForge.net: htdig-general
In cases where you know exactly which **extensions** you want to **exclude**, ... the **extension**, so it won"t get confused by **extensions** in **query** string parameters. ...
sourceforge.net/mailarchive/message.php?msg_id=7661422 - 15k - Cached - Similar pages

RE: **exclude**, include **query**
How about finding all files with these **extensions**; writing them to a file, then backing-up the ... Next by Thread:, Re: **exclude**, include **query**, Dave Markham ...
www.arcknowledge.com/gmane.comp.sysutils.backup.veritasbu/2006-03/msg00248.html - 16k - Cached - Similar pages

> Re: **exclude**, include **query**
> How about finding all files with these **extensions**, writing them to a file, > then backing-up ... Next by Date:, RE: **exclude**, include **query**, WEAVER, Simon ...
> www.arcknowledge.com/gmane.comp.sysutils.backup.veritasbu/2006-03/msg00264.html - 18k - Cached - Similar pages

[Paper] Concept-based Search System using Context-conserving ...
The information on the **query extension** is not stored in a static form, ... Therefore, the evaluator must **exclude** the words that are not context- conservable ...
www.actapress.com/PDFViewer.aspx?paperId=14081 - Similar pages

PHP: MySQL Functions - Manual
This simple example shows how to connect, execute a **query**, print resulting rows and disconnect from a MySQL database. Example 1328. MySQL **extension** overview ...
php.net/mysql - 157k - Feb 27, 2007 - Cached - Similar pages

How to **Exclude** Pages -- FreeFind.com
How to **Exclude** Pages ... prevents all files that end with the **extension** ".txt" from being ... Preventing links with **query** strings from being followed ...
www.freefind.com/library/howto/**exclude**/ - 28k - Cached - Similar pages

Google SOAP Search API
File Type Exclusion, Google -filetype:doc -filetype:pdf, The **query** prefix "-filetype:" filters the results to **exclude** documents with the **extension** specified ...
code.google.com/apis/soapsearch/reference.html - 100k - Cached - Similar pages

LOC reporting - MSDN Forums
You can slice the code churn measures with [Filename].[File **Extension**].[File **Extension**] and **exclude** ".vbproj" & ".wsdl" files from the results of your **query** ...

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	0	"6665682".pn.	US-PGPUB	OR	OFF	2007/03/01 11:31
L2	2	"6665682".pn.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/01 12:26
L3	0	"6363377".pn. and relational	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/01 12:26
L4	1	"6363377".pn. and database	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/01 12:48
L5	13754	707/3 or 707/4 or 707/5	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/01 12:49
L6	54	5 and (query near2 extension)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/01 12:49